ROUTING AND ACTION MEMORANDUM

ROUTING

TO: (1) Network Sciences Division (Iyer, Purush)

Report is available for review

(2) Proposal Files Proposal No.: 54231NSMUR.175

DESCRIPTION OF MATERIAL

CONTRACT OR GRANT NUMBER: W911NF-08-1-0242

INSTITUTION: University of Washington

PRINCIPAL INVESTIGATOR: Pedro Domingos

TYPE REPORT: Manuscript

DATE RECEIVED: 9/23/14 4:36PM

PERIOD COVERED: through

TITLE: Gradient Boosting for Conditional Random Fields

ACTION TAKEN BY DIVISION

(x) Report has been reviewed for technical sufficiency and IS [x] IS NOT [ ] satisfactory.

(x) Material has been given an OPSEC review and it has been determined to be non sensitive and, except for manuscripts and progress reports, suitable for public release.

Approved by SSL\PURUSH.S.IYER on 9/26/14 10:55AM

REPORT DOCUMENTATION PAGE

1

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 23-09-2014	2. REPORT TYPE Manuscript	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Gradient Boosting for Conditional Random Fields	5a. CONTRACT NUMBER W911NF-08-1-0242
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611103
6. AUTHORS Tianqi Chen, Sameer Singh, Carlos Guestrin	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Washington Office of Sponsored Programs 4333 Brooklyn Avenue NE Seattle, WA 98195 -9472	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 54231-NS-MUR.175

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

In this paper, we present a gradient boosting algorithm for tree-shaped conditional random fields (CRF). Conditional random fields are an important class of models for accurate structured prediction, but effective design of the feature functions is a major challenge when applying CRF models to real world data. Gradient boosting, which can induce and select functions, is a natural candidate solution for the problem. However, it is non-trivial to derive gradient boosting algorithms

15. SUBJECT TERMS

Conditional Random Fields, Gradient Boosting Algorithms, Hessian Matrices, Markov Chain Mixing Rate, Functional Space Optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Pedro Domingos
UU	UU	UU	UU		19b. TELEPHONE NUMBER 206-543-4229

Report Title

Gradient Boosting for Conditional Random Fields

ABSTRACT

In this paper, we present a gradient boosting algorithm for tree-shaped conditional random fields (CRF). Conditional random fields are an important class of models for accurate structured prediction, but effective design of the feature functions is a major challenge when applying CRF models to real world data. Gradient boosting, which can induce and select functions, is a natural candidate solution for the problem. However, it is non-trivial to derive gradient boosting algorithms for CRFs, due to the dense Hessian matrices introduced by variable dependencies. We address this challenge by deriving a Markov Chain mixing rate bound to quantify the dependencies, and introduce a gradient boosting algorithm that iteratively optimizes an adaptive upper bound of the objective function. The resulting algorithm induces and selects features for CRFs via functional space optimization, with provable convergence guarantees. Experimental results on three real world datasets demonstrate that the mixing rate based upper bound is effective for training CRFs with non-linear potentials.

Gradient Boosting for Conditional Random Fields

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we present a gradient boosting algorithm for tree-shaped conditional random fields (CRF). Conditional random fields are an important class of models for accurate structured prediction, but effective design of the feature functions is a major challenge when applying CRF models to real world data. Gradient boosting, which can induce and select functions, is a natural candidate solution for the problem. However, it is non-trivial to derive gradient boosting algorithms for CRFs, due to the dense Hessian matrices introduced by variable dependencies. We address this challenge by deriving a Markov Chain mixing rate bound to quantify the dependencies, and introduce a gradient boosting algorithm that iteratively optimizes an adaptive upper bound of the objective function. The resulting algorithm induces and selects features for CRFs via functional space optimization, with provable convergence guarantees. Experimental results on three real world datasets demonstrate that the mixing rate based upper bound is effective for training CRFs with non-linear potentials.

1 Introduction

Many problems in machine learning involve structured prediction, i.e. predicting a group of outputs that depends on each other. Conditional random fields [9] are among the most successful solutions to these problem. Variants of tree-shaped conditional random fields have been proposed and widely applied to structured prediction problems in domains such as natural language processing [9, 16], computer vision [7, 15] and bio-informatics [21]. As opposed to classification models that assume independent output variables, CRF models capture the dependency pattern between output and input via potential functions. Potential functions are usually defined using a linear combination of carefully engineered features of the input and the output variables. These feature functions are crucial for learning accurate models. Thus, it is important to ask whether we can induce arbitrary potential functions automatically (via functional space optimization), instead of manually crafting them and/or restricting them to linear combinations.

Gradient boosting [4], which performs additive training in functional spaces, is a natural candidate for this problem. Effective gradient boosting algorithms, such as LogitBoost and its variants [5, 11, 17], have been proposed for inducing feature functions for (independent) multi-class classification problems. The key ingredient in these methods is the effective use of second order information via diagonal approximation of Hessian matrices. Unfortunately, it is non-trivial to develop such boosting methods for CRFs, since variable interdependencies introduce dense Hessian matrices that make gradient boosting infeasible due to the computational complexity. Instead, existing boosting approaches either optimize approximate objectives [20, 12] or only take first order information into account when optimizing exact likelihood [1]. Unfortunately, the convergence of this method is guaranteed only with small step sizes.

In this paper, we present a novel gradient boosting algorithm for inducing non-linear feature functions for tree-shaped CRFs. The CRF training is performed via iteratively optimizing adaptive upper bounds of the loss function, to address the challenge of dense Hessians. The adaptive bounds, which are derived using Markov Chain mixing rates, measure the dependency between variables, and accordingly control the conservativeness of the updates. The resulting gradient boosting algorithm, which can be viewed as generalization of LogitBoost to structure prediction problems, optimizes the CRF objective with provable convergence guarantees. Experimental results on three real world datasets demonstrate the effectiveness and efficiency of our bound and the proposed algorithm.

2 Overview of Method

Model Formalization Given input x, a CRF model describes a distribution over the outputs as

$$P(y|x) = \frac{\exp(\Phi(y,x))}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(y',x))} \tag{1}$$

where $\mathcal{Y}$ is the set of possible output combinations, and $\Phi(y,x)$ captures the dependency between the input and output variables. The model Φ usually factorizes as a sum of unary and pairwise (edge) *potential functions* ϕ_i between individual output variables, which can be expressed as follows:

$$\Phi(y,x) = \sum_{i=1}^{m} \phi_i(x)\mu_i(y),\ \phi_i \in \mathcal{F}, \mu_i \in N \cup \mathcal{E} \quad \text{subject to } \phi_i = \phi_j \text{ for } (i,j) \in \mathcal{C}, \tag{2}$$

where $N = \{\mathbf{1}(y_t = k)\}, \mathcal{E} = \{\mathbf{1}(y_s = k_1, y_t = k_2)\}$ are the sets of indicator functions of each node and edge state. Each μ_i corresponds to a event $y_t = k$ or $y_s = k_1, y_t = k_2$ (depending on whether ϕ_i is node or edge potential). We use μ as short hand for $\mu(y)$ and view it as a vector of random variables. The family of all possible node and edge potential functions is $\mathcal{F} = \mathcal{F}_n \cup \mathcal{F}_e$, whose size could be infinite. $\mathcal{C}$ represents equivalence classes in different parts of the model that we use to capture parameter sharing, which is common in most applications of CRFs. In standard linear-chain CRFs with linear potentials, $\mathcal{F}$ contains linear functions of x, and $\mathcal{C}$ can be used to constrain the node and edge potential functions in different position to be the same. On the other hand, LogitBoost considers arbitrary $\mathcal{F}$, however constrains the model to contain only node potentials (there are no edge potentials). In this paper we are interested in arbitrary function families $\mathcal{F}$, and focus on tree-shaped $\mathcal{E}$ that allow exact inference of marginals.

Training Objective Using functions ϕ in Eq.(2) as the model parameters allows us to induce potential functions automatically through functional space optimization. In particular, we generalize the standard CRF objective over training data $\mathcal{D} = \{(y,x)\}$ to the following:

$$L(\phi) = \sum_{y,x \in \mathcal{D}} l(y,x,\phi) + \sum_{c} \Omega(\phi_c) = -\sum_{y,x \in D} \ln P(y|x) + \sum_{c} \Omega(\phi_c). \tag{3}$$

Here l is the negative log-likelihood function over each data point. $\sum_c \Omega(\phi_c)$ is a regularization term that measures the complexity of the learned function, defined as a sum over the equivalence class defined by $\mathcal{C}$. In standard CRFs, for example, Ω is often the square of L_2 norm of the parameter vector. This generalized objective function encourages us to select *predictive* (i.e. optimizes l) and *simple* (i.e. optimizes Ω) functions as potentials of a CRF.

Challenges for Function Learning Since the model parameters in this formulation are functions, Eq.(3) cannot be directly optimized using traditional optimization techniques. Instead, we train the model additively: at each iteration t, our proposed algorithm first searches over the functional space $\mathcal{F}$ to find functions $\delta = [\delta_1, \delta_2, \cdots, \delta_m]$ that optimize the objective function $L(\phi^{(t)} + \delta)$, and then adds them to the current model $\phi^{(t+1)} \leftarrow \phi^{(t)} + \delta$. However, due to the complex nature of the objective function, directly performing such a brute-force search requires a large amount of computation and is thus infeasible. In the same spirit as LogitBoost [5, 11] for multi-class prediction, we consider the second order Taylor expansion of the negative log-likelihood $l(y,x,\phi)$:

$$l(y,x,\phi+\delta) \simeq l(y,x,\phi) + \delta^T \mathbf{G}(y,x) + \frac{1}{2}\delta^T \mathbf{H}(y,x)\delta. \tag{4}$$

The gradient $\mathbf{G}$ and Hessian $\mathbf{H}$ in Eq.(4) are given by the following equation, where p_i and p_{ij} are short hand notations for $p_i \triangleq P(\mu_i = 1|x)$, $p_{ij} \triangleq P(\mu_i \mu_j = 1|x)$:

$$\mathbf{G}_i = \mu_i(y) - p_i, \quad \mathbf{H}_{ij} = p_{ij} - p_i p_j. \tag{5}$$

Note that Eq.(5) holds for all i, j pairs, including two special cases: (1) $\mathbf{H}_{ii} = p_i(1-p_i)$ when $i = j$, and (2) $\mathbf{H}_{ij} = -p_i p_j$ when μ_i and μ_j are mutual events. Intuitively, $\mathbf{H}_{ij}$ measures the correlation between two events, and is *nonzero* due to the dependencies in the CRF model. These dense elements of the Hessian make direct optimization of Eq. (4) still very costly. An existing approach to functional optimization for CRFs, presented in [1], resorts to first order approximation to the loss, and can only guarantee convergence when the step size is small. An alternative is to iteratively update one δ_i for $i \in \{1, \cdots, m\}$ at a time. This approach would require m inference steps per iteration, and is simply not applicable when the constraint $\mathcal{C}$ exists.

Our Approach In this paper, we consider an upper bound of Eq. (4) instead. The intuition behind this approach, which will be formalized in following sections, is as follows: each variable in the CRF depends weakly on variables that are "far" from it. This motivates the use of a diagonal upper bound of the Hessian to construct loss functions, given by the following Lemma.

Lemma 2.1. *Let $\mathcal{U}$ be a index set of potential functions we want to update, and let γ be a function that satisfies the following inequality*

$$\gamma_i(y,x)\mathbf{H}_{ii}(y,x) \geq \sum_{j\in\mathcal{U}} |\mathbf{H}_{ij}(y,x)| \tag{6}$$

Then for $\delta \in \{[\delta_1, \delta_2, \cdots, \delta_m] \mid \delta_i = 0$ for $i \notin \mathcal{U}\}$, the following inequality holds,

$$l(y,x,\phi+\delta) \leq l(y,x,\phi) + \sum_{i\in\mathcal{U}}[\delta_i \mathbf{G}_i(y,x) + \frac{1}{2}\gamma_i(y,x)\mathbf{H}_{ii}\delta_i^2(y,x)] + o(\delta^2). \tag{7}$$

The detailed proof of the lemma is given in supplementary material. For a given γ that satisfies the condition, we can iteratively optimize $\tilde{L}(\phi,\delta)$, which is an upper bound of $L(\phi)$, defined by

$$\tilde{L}(\phi,\delta) = L(\phi)+\sum_{i\in\mathcal{U}}[(\sum_{x,y\in\mathcal{D}}\mathbf{G}_i(y,x))\delta_i(y,x)+\frac{1}{2}(\sum_{x,y\in\mathcal{D}}\gamma_i(y,x)\mathbf{H}_{ii})\delta_i^2(y,x)+\Omega(\phi_i+\delta_i)-\Omega(\phi_i)]. \tag{8}$$

$\tilde{L}(\phi,\delta)$ is composed of $|\mathcal{U}|$ independent loss functions with a regularization term, and can be used to guide the common function search (such as regression tree learning). Iteratively optimizing $\tilde{L}$ will result in a gradient boosting algorithm that ensures the convergence of $L(\phi)$ (Proof in Section 4). Furthermore, the form of $\tilde{L}$ allows the search of δ_i for $i \in \mathcal{U}$ to be done *in parallel* for each equivalent class defined by $\mathcal{C}$, which gives us further computational benefits. In the next two sections, we will discuss how we can efficiently estimate γ when $\mathcal{U}$ is the index set of all node potentials, and when it is the index set of all edge potentials, using a mixing rate of Markov chain.

3 Upper Bound Derivation using a Markov Chain Mixing Rate

In this section, we will discuss how we can estimate γ when $\mathcal{U}$ is the index set of all node potentials, and the index set of all edge potentials. Conceptually, the choice of γ should be related to the interdependency of variables in the current model. When the variables in the model are independent from each other, γ should be small, and when the variables in the model have strong dependencies, γ should become larger. We want to *quantitatively* measure the dependencies in the CRF. Specifically, we will connect the dependency level to the mixing rate of a Markov chain defined by the conditional distribution of outputs on input $P(y|x)$. To begin with, we re-express the right side of Eq. (6) using total variation distance, defined by $\|P-Q\|_{tv} = \frac{1}{2}\sum_x |P(x)-Q(x)|$.

Lemma 3.1. *Let $\mathcal{U}$ correspond to the set of all node potentials $\mathcal{U} = \{j|\phi_j \in N\}$, assuming index i corresponds to the event $y_t = k$ (i.e. $\mu_i = \mathbf{1}(y_t = k)$), then*

$$\sum_{j\in\mathcal{U}} |\mathbf{H}_{ij}| = 2p_i \sum_s \|P(y_s|x, \mu_i = 1) - P(y_s|x)\|_{tv}. \tag{9}$$

Lemma 3.2. *Let $\mathcal{U}$ correspond to the set of all edge potentials $\mathcal{U} = \{j|\phi_j \in \mathcal{E}\}$, then*

$$\sum_{j\in\mathcal{U}} |\mathbf{H}_{ij}| = 2p_i \sum_{(s,v)\in\mathcal{E}} \|P(y_s, y_v|x, \mu_i = 1) - P(y_s, y_v|x)\|_{tv}.$$

Note that we abuse the notation slightly here, by using $\mathcal{E}$ to indicate the index set of edges in CRF.

The proof is a re-arrangement of terms, and is provided in the supplementary material. Intuitively, the total variation terms in Lemma 3.1 and 3.2 measure how dependent y_s is on the event $y_t = k$. When y_s is only weakly dependent on y_t, the distance will be small. The complexity of calculating Eq. (9) for all i is quadratic in the number of nodes, which is too expensive to be calculated directly for most applications. We need an algorithm that scales linearly with the number of nodes.

Total variation distance allows us to approach the problem in terms of dependencies between variables. Intuitively, we expect the dependencies between y_s and y_t to become smaller as we change s to get away from t. We formally state this in the following theorem:

Theorem 3.1. *Mixing rate bound for Markov chain in CRF*
Assume y_t, y_s and y_v form a Markov chain $y_t \to y_s \to y_v$, conditioned on x, *i.e. $P(y_v|y_s, y_t, x) = P(y_v|y_s, x)$ holds. Define $d(s,t,k) \triangleq \|P(y_s|x, y_t = k) - P(y_s|x)\|_{tv}$. Then, the total variation $d(v,t,k)$ can be bounded by*

$$d(v,t,k) \le [1 - \sum_j \min_i P(y_v = j|y_s = i, x)]d(s,t,k) \triangleq \alpha_{s,v} d(s,t,k). \tag{10}$$

Proof. Define notation: $M_{ij} \triangleq P(y_v = j|y_s = i, x), Q_j \triangleq \min_i M_{ij}$, then

$$\begin{aligned}
2d(v,t,k) &= \sum_j |P(y_v = j|y_t = k, x) - P(y_v = j|x)| \\
&= \sum_j |\sum_i M_{ij} P(y_s = i|y_t = k, x) - \sum_i M_{ij} P(y_s = i|x)| \\
&= \sum_j |\sum_i (M_{ij} - Q_j)[P(y_s = i|y_t = k, x) - P(y_s = i|x)]| \\
&\le \sum_j \sum_i (M_{ij} - Q_j)|P(y_s = i|y_t = k, x) - P(y_s = i|x)| \\
&= \sum_i (1 - \sum_j Q_j)|P(y_s = i|y_t = k, x) - P(y_s = i|x)| = 2\alpha_{s,v} d(s,t,k) \qquad \square
\end{aligned}$$

The derivation of Theorem 3.1 is inspired, in spirit, by the mixing rate bounds of time homogeneous Markov Chains [10][1]. Intuitively, Theorem 3.1 shows the dependency decays exponentially as s moves away from t. The following corollary holds as the direct consequence of the theorem.

Corollary 3.1. *Let $q = [q(1), q(2), \cdots q(n)]$ be the path sequence in $\mathcal{E}$ from t to s (i.e. $q(1) = t, q(n) = s$) then we can bound $d(t,s,k)$ using $d(t,t,k)$ times the decay ratio α along the path,*

$$d(s,t,k) \le \prod_i^{n-1} \alpha_{q(i),q(i+1)} d(t,t,k). \tag{11}$$

In the case when $\mathcal{E}$ is a chain, Corollary 3.1 simplifies to $d(s,t,k) \le \prod_{h=t}^{s-1} \alpha_{h,h+1} d(t,t,k)$ when $s > t$, and $d(s,t,k) \le \prod_{h=s+1}^{t} \alpha_{h,h-1} d(t,t,k)$ when $s < t$. An important property of Theorem 3.1 is that the position specific mixing rate $\alpha_{s,v}$ can be computed efficiently (complexity analysis in Sec 4). We still need to calculate $d(t,t,k)$, which is given by the following Lemma

Lemma 3.3. *Let M correspond to the index set of μ_i such that μ_i, μ_j are mutual to each other (i.e. $\mu_i \mu_j = 0$ for $i \ne j, i, j \in M$), and $\sum_{j \in M} P(\mu_i = j|x) = 1$. Then the following identity holds*

$$\frac{1}{2} \sum_{j \in M} |P(\mu_j = 1|\mu_i = 1, x) - P(\mu_j = 1|x)| = 1 - P(\mu_i = 1|x) \tag{12}$$

The proof is given in supplementary material. From Lemma 3.3, it follows that $d(t,t,k) = 1 - p_i$. We will make use of Lemma 3.3 and Corollary 3.1 to efficiently estimate γ in next section.

4 Gradient Boosting for CRF

In this section, we will present our gradient boosting algorithm. We will give estimation of γ for $\mathcal{U}$ to be the index set of all node potentials and edge potentials, given by the following two theorems.

[1] Note that our proof is actually for a time inhomogeneous Markov Chain.

Algorithm 1 Gradient Boosting for CRF

```
repeat
  for $\mathcal{U} \in \{N, \mathcal{E}\}$ do
    for $y, x \in \mathcal{D}$ in parallel do
      {inference of $p_i, \gamma_i$ are done using dynamic programming}
      Infer $\mathbf{G}_i(y,x) \leftarrow \mu_i(y) - p_i, \mathbf{H}_{ii}(y,x) \leftarrow p_i(1-p_i)$ for each $i \in \mathcal{U}$
      Infer $\gamma_i(y,x)$ using Theorem 4.1 and 4.2 for each $i \in \mathcal{U}$
    end for
    for $[c] \subset \mathcal{U}$ in parallel do
      {We use $[c]$ to enumerate over set of equivalent index defined by $\mathcal{C}$ in $\mathcal{U}$}
      $\delta_c \leftarrow \text{argmin}_{\delta \in \mathcal{F}_n} \Omega(\phi_i + \delta) + \sum_{i \in [c]} \sum_{y,x \in \mathcal{D}} \left[\mathbf{G}_i(y,x)\delta(y,x) + \gamma_i(y,x)\mathbf{H}_{ii}(y,x)\delta^2(y,x)\right]$
      $\phi_c \leftarrow \phi_c + \epsilon\delta_c$
    end for
  end for
until convergence
```

Theorem 4.1. *Let $\mathcal{U}$ be the index set of all node potentials, assume $\mu_i = \mathbf{1}(y_t = k)$, and define $\mathcal{Q}_t$ to be the set of all paths that start from t, then*

$$\gamma_i(y,x) = 2(1 + \sum_{q \in \mathcal{Q}_t} \sum_{s=2}^{len(q)} \prod_{i=1}^{s-1} \alpha_{q(i),q(i+1)}) \tag{13}$$

satisfies Eq.(6). We use $\text{len}(q)$ as the length of the path, α is defined in Theorem 3.1.

Theorem 4.2. *Let $\mathcal{U}$ be the index the set of all edge potentials, assume $\mu_i = \mathbf{1}(y_t = k_1, y_{t+1} = k_2)$, and define $\mathcal{Q}_{t,t+1}$ to be the set of all path that start from t and $t+1$ and do not cross $(t, t+1)$, then*

$$\gamma_i(y,x) = 2(1 + \sum_{q \in \mathcal{Q}_{t,t+1}} (1 + \sum_{s=2}^{len(q)} \prod_{i=1}^{s-1} \alpha_{q(i),q(i+1)})) \tag{14}$$

satisfies Eq.(6), with the same definition of $\text{len}(q)$ and α as in Theorem 4.1.

Both theorems can be proved by using Corollary 3.1 and Lemma 3.3 to bound the total variation distance. We give the detailed proof in supplementary material. Based on Theorem 4.1 and 4.2, we can get an efficient gradient boosting algorithm for CRF (GBCRF), which is presented in Algorithm 1. Here ϵ is a shrinkage term used to control overfitting. Our algorithm adaptively estimates γ via the mixing rate calculation at each iteration. At the beginning of training, where each node variable is independent from each other, we will have a γ that is close to 2 (and thus the updates are aggressive). γ increases as the variables become dependent on each other (inducing more conservative updates).

The calculation of γ can be performed using dynamic programming. To explain the idea more clearly, let us consider the case when $\mathcal{E}$ is a chain. In this case, Eq. (13) specializes into a calculation of $\beta_t^+ \triangleq \sum_{s=t}^{n} \prod_{i=t}^{s-1} \alpha_{i,i+1}$ and $\beta_t^- \triangleq \sum_{s=1}^{t} \prod_{i=s+1}^{t} \alpha_{i,i-1}$, and both can be calculated efficiently using the following recursive formula

$$\beta_t^+ = \alpha_{t,t+1}(1 + \beta_{t+1}^+), \ \beta_t^- = \alpha_{t,t-1}(1 + \beta_{t-1}^-). \tag{15}$$

Similarly, we can use dynamic programming for any tree-shaped $\mathcal{E}$ (using up-down recursion). A direct consequence of Theorem 4.1 is that we can bound the loss using estimation by number of nodes in CRF. Though this bound is usually worse than the bound using mixing rate.

Corollary 4.1. *When $\mathcal{U}$ is the index set of node potentials , $\gamma_i = 2n$ satisfies Eq. (6), where n is number of nodes in CRF.*

Relation to LogitBoost Our algorithm can be viewed as a generalization of multi-class classification using LogitBoost [5]. When the variables in each position are independent (no edge potentials), the estimation of γ equals 2, and our algorithm becomes identical to LogitBoost. When the variables are dependent on each other, which is common in structured prediction, our model estimates the dependency level via the Markov Chain mixing rate to guide the boosting objective in each iteration.

Time Complexity The time complexity for the gradient boosting statistics collection in Algorithm 1 is $O(|\mathcal{D}|nK^2)$, where K is the number of states in each node and n is the average number of nodes (e.g. length of sequence) in each instance. This is due to the fact that estimation of γ can be done in $O(|\mathcal{D}|nK^2)$ complexity, using a dynamic programming algorithm. This complexity is *same as* the complexity for traditional training methods for linear CRF. The time complexity of entire algorithm is $O(|\mathcal{D}|nK^2 + g(|\mathcal{D}|, n))$, where $g(|\mathcal{D}|, n)$ is cost of function learning given the statistics. For learning trees, the complexity of function learning is usually $O\left(|\mathcal{D}|n\log(|\mathcal{D}|n)\right)$. Thus our approach extends CRFs to non-linearity with only an additional log factor.

Convergence Analysis In this section, we analyze the convergence of our algorithm. The advantage of our method is that it makes use of second order information, and guarantees convergence.

Theorem 4.3. *$L(\phi)$ converges with the procedure described by Algorithm 1 for $\epsilon \leq 1$.*

Proof. During each iteration, assume δ^* is the function that optimizes $\tilde{L}(\phi, \delta)$ defined in Eq. (8),

$$L(\phi + \epsilon\delta^*) \leq \tilde{L}(\phi, \epsilon\delta^*) \leq \tilde{L}(\phi, \mathbf{0}) = L(\phi) \tag{16}$$

Thus the loss function L decreases after each boosting step, and the algorithm converges to a minima (possibly local minima when $\mathcal{F}$ is nonlinear) of L. □

5 Related Work

Conditional random fields [9] are among the most successful solutions to structured prediction problems. Variants of conditional random fields have been proposed and widely applied for structured prediction in domains such as natural language processing [9, 16], computer vision [7, 15] and bioinformatics [21]. Most popular instantiations assume linear potential functions and improve the performance by carefully engineering features. Our work focuses on learning probabilistic models for tree-shaped CRFs with nonlinear potential functions. When there are loops in the CRF and inference is intractable, relaxation of the objective can be done to use approximate inference and learning [6, 13, 3]. A similar dependency based term is also used in the approximate inference [13, 3], but is usually set be a constant value across all instances and training iterations. As a future work, it would be interesting to explore whether our adaptive Markov Chain mixing rate bound can be applied to this more general setting.

Gradient boosting [4], which performs additive optimization in the functional space, has been successfully applied to classification problems that assume independent outputs conditioned on the input [5]. Most existing attempts to "boost" CRF models optimize approximate objectives [20, 12]. TreeCRF algorithm [1] is similar to our approach in that it directly optimizes the log-likelihood function defined using non-linear potential functions, however they only take first order information into account during optimization, requiring a decreasing step size. On the other hand, our method makes use of second order information, and guarantees convergence with fixed step size. Our method can also be viewed as a generalization of LogitBoost [5] for CRF. It is worth noting that the recent improvements of LogitBoost, which make use of adaptive base function [11, 17], can also potentially be combined with our method to make further improvements.

6 Experiments

In this section, we evaluate our method on named entity recognition, hand written character recognition, and protein secondary structure prediction. We compare the following methods: (1) **GBCRF** is the proposed method in this paper. We set $\mathcal{F}_n$ to be a set of regression trees, and $\mathcal{F}_e$ to be linear functions of basic transition features between states; (2) **LogitBoost** is a gradient boosting method for multi-class classification [5] that does not support the dependencies between outputs; (3) **TreeCRF** is a gradient boosting method that only takes *first-order* information [1]. We use the same family of edge and node potentials as GBCRF; (4) **Linear CRF** is the standard CRF model with linear edge and node potentials [9]. For all the methods, the training parameters are selected using a validation set or cross validation, depending on the specific setup of each dataset.

Named Entity Recognition We first test our methods on the natural language task of named entity recognition (NER) using the CoNLL-2003 shared task benchmark dataset [19]. The dataset contains around 20K sentences, and defines a standard split into 14K as training set, 3.3K as validation

Table 1: F1 Measure of Name Entity Recognition on CoNLL-2003 Dataset. We use subscript val to denote validation set, and subscript $test$ to denote test set.

	Word Embedding only		Word Features + Embedding	
Method	F1_{val}	F1_{test}	F1_{val}	F1_{test}
Linear CRF	0.8452	0.7943	0.8952	0.8475
LogitBoost	0.8532	0.7887	0.8717	0.8197
TreeCRF	0.8630	0.8060	0.8846	0.8399
GBCRF	0.8801	0.8269	0.9015	0.8635

Table 2: Cross Validation Error on Handwritten Character Recognition Dataset.

Method	Error
Linear CRF	0.1292 ± 0.0080
LogitBoost	0.0967 ± 0.0049
TreeCRF	0.0699 ± 0.0040
GBCRF	0.0464 ± 0.0027
NeuroCRF (Do et al.[2])	0.0444

Table 3: Predictive Q8 Accuracy on Protein Secondary Structure Dataset.

Method	Accuracy
Linear CRF	0.614
LogitBoost	0.710
TreeCRF	0.718
GBCRF	0.722
SC-GSN (Zhou et al.[22])	0.711
SC-GSN with “kick-start” ([22])	0.721

set (also called development set), and 3.5K sentences as test set. Traditional approaches for NER involve a lot of time-consuming feature engineering that requires domain expertise, and build a Linear CRF over these features. Instead, in our experiment, we explore whether it is possible to perform minimal feature engineering, and use a representation learned from data for prediction.

Specifically, we take the word embedding vectors from Mikolov et.al [14], which is learned from Google news corpus, and train the models on this representation. In this setting , each word is represented by a 300 dimensional vector that captures the “semantics” of the word. For each position in the sentence, we take the embedding vector of the previous, current, and next word as input to node potential function. We call this setting “word embedding only”. We further perform minimal feature engineering to *only* generate the unigram features (word, postag and case pattern of current word). We use these basic features to train a weak linear model, then use additive training to boost the base model using the word embedding representation. We call this setting “word feature+embedding”.

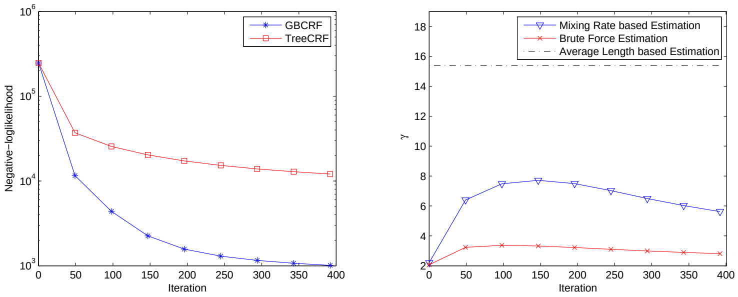
The results of token-wise F1 evaluation for these models are shown in Table 1. From the result, we see that GBCRF works better than Linear CRF in both settings. The gap between LogitBoost and GBCRF indicates the importance of introducing edge potentials to this problem. We also find that taking second order information into account helps us obtain a more accurate model.

Handwriting Character Recognition We also evaluate our method on a handwriting recognition dataset[2]. The dataset consists of 6877 words and corresponds to about 52 thousand handwritten characters [8, 18], each represented by a binary pixel vector of 128 dimensions, and belonging to one of 26 alphabets. The dataset is randomly split into 10 folds for cross validation. We train the models on 9 folds, test on 1 fold, and use the cross validation error to compare the methods.

The experiment results are shown in Table 2. Both our method and TreeCRF outperforms CRF with linear potential functions, which indicates the effectiveness of introducing a non-linear potential function into the CRF on this dataset. The gap between LogitBoost and models that consider dependencies indicate the importance of incorporating structure information of the outputs into the model. Our results are also comparable to NeuroCRF [2], which uses a deep neural network as a potential function whose weights are initialized by Restricted Boltzmann Machines.

Protein Secondary Structure Prediction We also conduct an experiment on protein secondary structure prediction. The task is to predict 8-state secondary structure labels for a given amino-acid sequence of a protein. We use the protein secondary structure data-set recently introduced by Zhou et al.[22], which is the largest publicly available protein secondary structure prediction dataset. The dataset contains 6128 proteins, with average sequence length around 208. We use exactly the same features and data split step as [22]. The resulting data set contains 5600 sequences as training set, 256 sequences as validation set and 272 sequences as test set. Each position of the protein sequence contains 46 dimension features (22 for PSSM, 22 for sequence and 2 for terminals) for prediction.

[2] http://www.seas.upenn.edu/ taskar/ocr/



(a) Evolution of Negative-loglikelihood (b) Evolution of average γ estimation

Figure 1: Convergence of GBCRF on hand written character dataset. (a) Convergence comparison between GBCRF and TreeCRF, with shrinkage rate of both algorithm set to 1. GBCRF converges faster than TreeCRF; (b) Evolution of different γ estimations on the CRF model in each round based on 500 sequences. Mixing rate based estimation has the same trend as brute force estimation, and provides a tighter estimation than the length based estimation.

To train the models, we take the concatenation of feature vectors within 3 positions of the target position as input to the node potential, resulting in 322 input features in each position.

The performance of the model is measured by the accuracy of predictions on the test set (denoted as Q8). We train the models with parameters discovered using the validation set, and report the results in Table 3. From the table, we find that using trees as potential functions leads to better performance than restricting the model to using linear functions. Our results are comparable to the state-of-art result in this dataset, produced by Zhou et al. [22] (SC-GSN-3layer). The result is generated by a deep convolutional generative stochastic network model to perform secondary structure label prediction, optimized with a "kick-start" initialization scheme.

Convergence of GBCRF We further analyze the convergence of GBCRF on the handwritten character dataset. We plot the convergence of negative log-likelihood function of GBCRF and TreeCRF in Fig. 1(a). We find that GBCRF converges faster than TreeCRF, demonstrating that taking second order information into account not only gives theoretical guarantee of convergence, but also helps the method to converge faster in practice.

We also investigate the tightness of γ estimation. Figure 1(b) gives the average of different γ estimations on models trained by GBCRF in each round. Mixing rate based estimation is the method proposed in this paper. We perform Brute Force estimation to compute γ exactly using Eq. (9); the complexity of this estimation is quadratic in the number of nodes and outputs in the CRF, and hence cannot be used for most real-world sequences. Average Length based estimation is a naive estimation using 2 times number of nodes in CRF, and provides an valid estimation of γ since it upper bounds Eq. (13), as we show in Corollary 4.1. We restrict this evaluation to the shortest 500 sequences, due to the computation cost of brute force estimation. From the figure, we find that mixing rate based estimation exhibits the same trend as the brute force estimation, and is at most 2.3 times higher than the brute force estimation. Further, the mixing rate based bound is consistently lower than the fixed bound computed by the length based estimation. These results indicate that our mixing rate based estimation captures the changes in the dependencies in the model during training correctly. Hence our proposed mixing rate based approach is indeed useful to estimate γ efficiently.

7 Conclusion

In this paper, we present novel gradient boosting algorithm for CRF. It is non-trival to design an effective gradient boosting for CRF, mainly due to the dense Hessian matrices introduced by variable interdependency. To solve the problem, we make use of a Markov Chain mixing rate to derive an efficiently computable adaptive upper bound of the loss function, and construct a gradient boosting algorithm that iteratively optimizes the bound. The resulting algorithm can be viewed as a generalization of LogitBoost to CRF, thus introducing non-linearity in CRFs at only a log factor cost. Experimental results demonstrate that our method is both efficient and effective. As future work, it is interesting to explore whether we can generalize the result to loopy models.

References

[1] T. Dietterich, G. Hao, and A. Ashenfelter. Gradient tree boosting for training conditional random fields. *Journal of Machine Learning Research*, 9:2113–2139, 2008.

[2] T. Do and T. Artieres. Neural conditional random fields. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, 2010.

[3] J. Domke. Structured learning via logistic regression. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 647–655. 2013.

[4] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[5] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[6] T. Hazan and R. Urtasun. Efficient learning of structured predictors in general graphical models. *arXiv*, 1210.2346, 2012.

[7] X. He, R. S. Zemel, and M. A. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'04, pages 695–703, 2004.

[8] R. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Cambridge, MA, USA, 1995.

[9] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*, pages 282–289, 2001.

[10] D. Levin, Y. Peres, and E. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

[11] P. Li. Robust Logitboost and adaptive base class (ABC) Logitboost. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI'10)*, pages 302–311, 2010.

[12] L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 2007.

[13] O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. Learning efficiently with approximate inference via dual losses. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 783–790, 2010.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 3111–3119. 2013.

[15] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 1097–1104.

[16] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, pages 134–141, 2003.

[17] P. Sun, J. Zhou, and M. D. Reid. AOSO-LogitBoost: Adaptive one-vs-one Logitboost for multi-class problem. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 1087–1094, 2012.

[18] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS'04)*, pages 25–32, 2004.

[19] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[20] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 1401–1408.

[21] J. Vinson, D. Decaprio, M. Pearson, S.Luoma, and J. Galagan. Comparative gene prediction using conditional random fields. In *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 1441–1448. 2007.

[22] J. Zhou and O. Troyanskaya. Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. In *Proceedings of the 30th International Conference on Machine Learning (ICML'14)*, volume 32, pages 745–753, 2014.

Supplementary Material

A Proof for Lemma 2.1

Proof. The following inequality holds for γ that satisfies the condition

$$\sum_{i\in\mathcal{U}}\gamma_i\mathbf{H}_{ii}\delta_i^2 \geq \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{U}}|\mathbf{H}_{ij}|\delta_i^2 = \frac{1}{2}\sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{U}}|\mathbf{H}_{ij}|(\delta_i^2+\delta_j^2) \geq \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{U}}|\mathbf{H}_{ij}|\delta_i\delta_j$$

Applying it to Talyor expansion in Eq (4), we have

$$\begin{aligned} l(y,x,\phi+\delta) &= l(y,x,\phi)+\sum_{i\in\mathcal{U}}\delta_i\mathbf{G}_i(y,x)+\frac{1}{2}\sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{U}}|\mathbf{H}_{ij}|\delta_i\delta_j+o(\delta^2) \\ &\leq l(y,x,\phi)+\sum_{i\in\mathcal{U}}\delta_i\mathbf{G}_i(y,x)+\frac{1}{2}\sum_{i\in\mathcal{U}}\gamma_i\mathbf{H}_{ii}\delta_i^2+o(\delta^2). \end{aligned}$$

□

B Proof for Lemma 3.3

Proof. Taking the fact that μ_i and μ_j are mutual for $j \neq i$, we have

$$\begin{aligned} &\sum_{j\in M}|P(\mu_j=1|\mu_i=1,x)-P(\mu_j=1|x)| \\ =&|P(\mu_i=1|\mu_i=1,x)-P(\mu_i=1|x)|+\sum_{j\neq i}|P(\mu_j=1|\mu_i=1,x)-P(\mu_j=1|x)| \\ =&|1-P(\mu_i=1|x)|+\sum_{j\neq i}|0-P(\mu_j=1|x)| \\ =&(1-P(\mu_i=1|x))+\sum_{j\neq i}P(\mu_j=1|x) \\ =&2(1-P(\mu_i=1|x)) \end{aligned}$$

□

C Proof for Lemma 3.1 and 3.2

Proof. The proof is exactly the same for both node and potential case, we present the proof for $\mathcal{U}$ to be all node potentials here. Recall the definition of $\mathbf{H}$: $\mathbf{H}_{ij}=p_{ij}$. Note that p_i and p_{ij} are short hand notations for $p_i \triangleq P(\mu_i=1|x)$, $p_{ij} \triangleq P(\mu_i\mu_j=1|x)$, we have

$$\begin{aligned} \frac{1}{2p_i}\sum_{j\in\mathcal{U}}|\mathbf{H}_{ij}| &= \sum_j |p_{ij}/p_i-p_j| \\ &= \sum_{j\in\mathcal{U}}|P(\mu_j=1|\mu_i=1,x)-P(\mu_j=1|x)| \\ &= \sum_{s,k'}|P(y_s=k'|y_t=k,x)-P(y_s=k'|x)| \\ &= \sum_s \|P(y_s|x,y_t=k)-P(y_s|x)\|_{tv} \end{aligned}$$

□

D Proof for Theorem 4.1 and 4.2

Proof. **Proof for Theorem 4.1** Basically, we want to bound the total variation distance given by Eq. (9) in Lemma 3.1,

$$\begin{aligned} 2p_i \sum_s \|P(y_s|x, y_s = k) - P(y_s|x)\|_{tv} = 2p_i \sum_s d(s, t, k) \\ = 2p_i[d(t, t, k) + \sum_{q \in \mathcal{Q}_t} \sum_{s=2}^{len(q)} d(q(s), t, k)] \\ \leq 2p_i[d(t, t, k) + \sum_{q \in \mathcal{Q}_t} \sum_{s=2}^{len(q)} d(t, t, k) \prod_{i=1}^{s-1} \alpha_{q(i),q(i+1)}] \\ = 2p_i(1 - p_i)[1 + \sum_{q \in \mathcal{Q}_t} \sum_{s=2}^{len(q)} \prod_{i=1}^{s-1} \alpha_{q(i),q(i+1)}] \end{aligned}$$

Here the inequality is given by Corollary 3.1 ($d(q(s), t, k) \leq d(t, t, k) \prod_{i=1}^{s-1} \alpha_{q(i),q(i+1)}$), and last equality is given by Lemma 3.3 ($d(t, t, k) = 1 - p_i$). Recall that $\mathbf{H}_{ii} = p_i(1 - p_i)$, we have proved Theorem 4.1. □

Proof. **Proof for Theorem 4.2** In this proof, we will reduce the total variation distance between joint distribution of edge states into total variation distance of marginal distribution over nodes, as in Theorem 4.1. Assume in edge pairs are (y_t, y_{t+1}), (y_s, y_{s+1}) , and y_s is closer to y_{t+1} (without loss of generality), then

$$P(y_s, y_{s+1}|y_t, y_{t+1}, x) = P(y_{s+1}|y_s, x)P(y_s|y_{t+1}, x)$$

We can convert total variation by

$$\begin{aligned} \|P(y_s, y_{s+1}|y_t, y_{t+1}, x) - P(y_s, y_{s+1}|x)\|_{tv} = \sum_{y_s, y_{s+1}} |P(y_s, y_{s+1}|y_t, y_{t+1}, x) - P(y_s, y_{s+1}|x)| \\ = \sum_{y_s, y_{s+1}} P(y_{s+1}|y_s, x)|P(y_s|y_{t+1}, x) - P(y_s|x)| \\ = \sum_{y_s} |P(y_s|y_{t+1}, x) - P(y_s|x)| \\ = \|P(y_s|y_{t+1}, x) - P(y_s|x)\|_{tv} \end{aligned} \tag{17}$$

Now the case become same as node potential, we can make use of Corollary 3.1 bound the total variation. Specifically, let $q \in \mathcal{Q}_{t,t+1}$ (i.e. $q(1) \in \{t, t+1\}, q(i) \notin \{t, t+1\}$ for $i > 1$)

$$\begin{aligned} &\sum_{i=1}^{len(q)} \|P(y_{q(i)}, y_{q(i+1)}|y_t = k_t, y_{t+1} = k_{t+1}, x) - P(y_{q(i)}, y_{q(i+1)}|x)\|_{tv} \\ =& \sum_{i=1}^{len(q)} \|P(y_{q(i)}|y_{q(1)} = k_{q(1)}, x) - P(y_{q(i)}|x)\|_{tv} \\ \leq& \|P(y_{q(1)}|y_{q(1)} = k_{q(1)}, x) - P(y_{q(1)}|x)\|_{tv} + \sum_{i=2}^{len(q)} \|P(y_{q(1)}|y_{q(1)} = k_{q(1)}, x) - P(y_{q(1)}|x)\|_{tv} \prod_{j=1}^{i-1} \alpha_{q(j),q(j+1)} \\ =& [1 - P(y_{q(1)} = k_{q(1)}|x)](1 + \sum_{i=2}^{len(q)} \prod_{j=1}^{i-1} \alpha_{q(j),q(j+1)}) \\ \leq& [1 - P(y_t = k_t, y_{t+1} = k_{t+1}|x)](1 + \sum_{i=2}^{len(q)} \prod_{j=1}^{i-1} \alpha_{q(j),q(j+1)}). \end{aligned}$$

Here the first inequality is due to Corollary 3.1. Summing the results over all $q \in \mathcal{Q}_{t,t+1}$ will give us Eq. (14). □